苏 州 大 学 实 验 报 告

院、系	计算机学院	年级专业	软件工程	姓名	朱金涛	学号	2327406014
课程名称	操作系统课程实践					成绩	
指导教师	王红玲	同组实验者	无		实验日期	2025 年 12 月 1 日	

实 验 名 称　　块设备驱动程序编写

一、 实验目的

- 了解 Linux 块设备管理机制。
- 学习块设备的基本管理方法。
- 学会编写简单的块设备驱动程序。

二、 实验内容

编写一个简单的块设备驱动程序，实现一套内存中的虚拟磁盘驱动器，并通过实际操作验证块设备驱动是否可以正常工作。

（1） 编写设备驱动源程序，即编写内核模块文件 simp_blkdev.c 和 Makefile 文件。

（2） 使用 make 命令编译驱动模块。

（3） 使用 insmod 命令安装驱动模块。

（4） 使用 lsblk 命令列出当前的块设备信息。

（5） 格式化设备 simp_blkdev。

（6） 创建挂载点并挂载块设备。

（7） 查看模块使用情况，会发现模块已被调用。

（8） 对块设备驱动进行调用测试。

（9） 取消挂载，查看模块调用结果。

（10） 使用 rmmod 命令卸载模块。

三、 实验步骤和结果

1. 编写驱动程序与 Makefile

首先，我编写了块设备驱动源程序 simp_blkdev.c 和编译规则文件 Makefile。在编写过程中，考虑到我使用的 Linux 6.17.2 内核版本较新，块设备层 API 发生了重构，我将头文件调整为 <linux/blkdev.h>，移除了过时的 BLK_MQ_F_SHOULD_MERGE 标志位，并为 blk_mq_alloc_disk 函数补全了缺失的参数，以确保代码能适配当前内核。

➢ 核心代码片段 1：请求处理函数（数据读写逻辑）

这段代码展示了驱动程序如何解析系统发来的请求，并通过 memcpy 在内存中完成读写操作。

```c
/* 核心逻辑：处理 I/O 请求 */
static blk_status_t simp_blkdev_queue_rq(struct blk_mq_hw_ctx *hctx,
                                         const struct blk_mq_queue_data *bd)
{
    struct request *rq = bd->rq;
    struct bio_vec bvec;
    struct req_iterator iter;
    loff_t pos = blk_rq_pos(rq) << 9; // 获取当前请求的扇区偏移量并转为字节
    void *buffer;

    blk_mq_start_request(rq); // 告诉内核开始处理请求

    // 遍历请求中的所有数据段
    rq_for_each_segment(bvec, rq, iter) {
        size_t len = bvec.bv_len;
        // 获取数据段在内存中的虚拟地址
        void *d_ptr = page_address(bvec.bv_page) + bvec.bv_offset;

        // 简单的读写逻辑：直接在申请的内存 (simp_blkdev_data) 上进行 memcpy
        buffer = simp_blkdev_data + pos;
        if (rq_data_dir(rq) == WRITE) {
            memcpy(buffer, d_ptr, len); // 写操作：缓冲区 -> 虚拟磁盘
        } else {
            memcpy(d_ptr, buffer, len); // 读操作：虚拟磁盘 -> 缓冲区
        }
        pos += len;
    }

    blk_mq_end_request(rq, BLK_STS_OK); // 告诉内核请求处理完成
    return BLK_STS_OK;
}
```

➢ 核心代码片段 2：模块初始化（设备注册逻辑）：

这段代码展示了如何申请 32MB 内存，并将其注册为一个标准的 Linux 块设备。

```c
/* 核心逻辑：模块初始化与设备注册 */
static int __init simp_blkdev_init(void)
{
    // 1. 申请 32MB 内存作为虚拟磁盘的存储空间
    simp_blkdev_data = vmalloc(SIMP_BLKDEV_BYTES);
    if (!simp_blkdev_data)
        return -ENOMEM;

    // 2. 向内核注册块设备，获取主设备号
    simp_blkdev_major = register_blkdev(0, SIMP_BLKDEV_DISKNAME);

    // 3. 初始化多队列标签集合 (blk-mq 架构的核心配置)
    tag_set.ops = &simp_blkdev_mq_ops; // 绑定上面的请求处理函数
    tag_set.nr_hw_queues = 1;          // 硬件队列数量
    tag_set.queue_depth = 128;         // 队列深度
    tag_set.numa_node = NUMA_NO_NODE;
    tag_set.cmd_size = 0;
    blk_mq_alloc_tag_set(&tag_set);

    // 4. 分配并初始化 gendisk 结构体 (适配 Linux 6.x 内核)
    // 注意：此处补全了 blk_mq_alloc_disk 缺少的参数以适配新内核
    simp_blkdev_disk = blk_mq_alloc_disk(&tag_set, NULL, NULL);

    // 5. 设置磁盘基本参数
    simp_blkdev_disk->major = simp_blkdev_major;
    simp_blkdev_disk->first_minor = 0;
    snprintf(simp_blkdev_disk->disk_name, 32, SIMP_BLKDEV_DISKNAME);
    set_capacity(simp_blkdev_disk, SIMP_BLKDEV_BYTES >> 9); // 设置容量(扇区数)

    // 6. 激活磁盘，使其对系统可见
    add_disk(simp_blkdev_disk);

    return 0;
}
```

2. 编译驱动模块

在终端中执行 **make** 命令对源码进行编译。编译系统根据 Makefile 的规则，调用当前内核的构建工具链，将 C 源码编译链接为可加载的内核模块文件 simp_blkdev.ko。经过针对新内核 API 的适配修改后，编译过程无报错顺利完成。

➢ 编译成功:

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ make
make -C /lib/modules/6.17.014Jintao/build M=/home/tt/下载/2327406014_朱金涛_实验十_源码 modules
make[1]: 进入目录“/home/tt/下载/linux-6.17.2”
make[2]: 进入目录“/home/tt/下载/2327406014_朱金涛_实验十_源码”
  CC [M]  simp_blkdev.o
  MODPOST Module.symvers
  CC [M]  simp_blkdev.mod.o
  CC [M]  .module-common.o
  LD [M]  simp_blkdev.ko
  BTF [M] simp_blkdev.ko
make[2]: 离开目录“/home/tt/下载/2327406014_朱金涛_实验十_源码”
make[1]: 离开目录“/home/tt/下载/linux-6.17.2”
```

3. 安装模块并查看设备信息

使用 sudo insmod simp_blkdev.ko 命令将编译好的驱动模块**加载到内核**中。

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo insmod simp_blkdev.ko
[sudo] tt 的密码:
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo dmesg | tail
[  228.639764] workqueue: blk_mq_run_work_fn hogged CPU for >10000us 7 times, consider switching to WQ_UNBOUND
[  402.719911] workqueue: psi_avgs_work hogged CPU for >10000us 4 times, consider switching to WQ_UNBOUND
[  404.105419] loop17: detected capacity change from 0 to 151456
[  406.179533] workqueue: psi_avgs_work hogged CPU for >10000us 5 times, consider switching to WQ_UNBOUND
[  485.059773] workqueue: psi_avgs_work hogged CPU for >10000us 7 times, consider switching to WQ_UNBOUND
[  707.279423] simp_blkdev: loading out-of-tree module taints kernel.
[  707.280232] simp_blkdev: module verification failed: signature and/or required key missing - tainting kernel
[  707.417008] Simp Block Device Driver Loaded
[  709.366176] kauditd_printk_skb: 149 callbacks suppressed
[  709.366189] audit: type=1400 audit(1764572440.593:161): apparmor="DENIED" operation="open" class="file" profile="snap
.firmware-updater.firmware-notifier" name="/proc/sys/vm/max_map_count" pid=5223 comm="firmware-notifi" requested_mask="r
" denied_mask="r" fsuid=1000 ouid=0
```

查看日志可以得知驱动模块已经成功加载。

随后，使用 lsblk 命令查看系统当前的**块设备列表**。结果显示系统中出现了一个名为 simp_blkdev 的设备，其容量为 32MB，这表明我的虚拟内存磁盘驱动已成功注册并被系统识别。

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ lsblk
loop0         7:0    0     4K  1 loop /snap/bare/5
loop1         7:1    0 330.2M  1 loop /snap/code/211
loop2         7:2    0  63.8M  1 loop /snap/core20/2669
loop3         7:3    0  63.8M  1 loop /snap/core20/2682
loop5         7:5    0  73.9M  1 loop /snap/core22/2139
loop6         7:6    0 248.8M  1 loop /snap/firefox/7024
loop7         7:7    0 249.1M  1 loop /snap/firefox/7177
loop8         7:8    0  11.1M  1 loop /snap/firmware-updater/167
loop9         7:9    0  18.5M  1 loop /snap/firmware-updater/210
loop10        7:10   0 516.2M  1 loop /snap/gnome-42-2204/226
loop11        7:11   0   516M  1 loop /snap/gnome-42-2204/202
loop12        7:12   0  91.7M  1 loop /snap/gtk-common-themes/1535
loop13        7:13   0  50.8M  1 loop /snap/snapd/25202
loop14        7:14   0  10.8M  1 loop /snap/snap-store/1270
loop15        7:15   0  50.9M  1 loop /snap/snapd/25577
loop16        7:16   0   576K  1 loop /snap/snapd-desktop-integration/315
loop17        7:17   0    74M  1 loop /snap/core22/2163
sda           8:0    0    80G  0 disk
├─sda1        8:1    0     1M  0 part
└─sda2        8:2    0    80G  0 part /
sr0          11:0    1  95.3M  0 rom  /media/tt/CDROM
sr1          11:1    1   5.9G  0 rom  /media/tt/Ubuntu 24.04.3 LTS amd64
simp_blkdev 251:0    0    32M  0 disk
```

4. 格式化虚拟块设备

为了能在该设备上存储文件，使用 sudo mkfs.ext4 /dev/simp_blkdev 命令对其进行**格式化**。该操作在我们的虚拟磁盘上构建了 ext4 文件系统，使其具备了文件存储和管理的能力。

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo mkfs.ext4 /dev/simp_blkdev
mke2fs 1.47.0 (5-Feb-2023)
创建含有 8192 个块（每块 4k）和 8192 个 inode 的文件系统

正在分配组表： 完成
正在写入 inode表： 完成
创建日志（1024 个块）： 完成
写入超级块和文件系统账户统计信息： 已完成
```

5. 挂载设备与读写测试

我创建了一个挂载点 /mnt/temp_disk，并使用 mount 命令将虚拟磁盘挂载到该目录。

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo mkdir -p /mnt/temp_disk
sudo mount /dev/simp_blkdev /mnt/temp_disk
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ df -h
文件系统          大小  已用  可用 已用% 挂载点
tmpfs             1.3G  2.1M  1.3G    1% /run
/dev/sda2          79G   58G   17G   78% /
tmpfs             6.4G   46M  6.3G    1% /dev/shm
tmpfs             5.0M  8.0K  5.0M    1% /run/lock
tmpfs             1.3G  116K  1.3G    1% /run/user/1000
/dev/sr1          6.0G  6.0G     0  100% /media/tt/Ubuntu 24.04.3 LTS amd64
/dev/sr0           96M   96M     0  100% /media/tt/CDROM
/dev/simp_blkdev   26M   24K   24M    1% /mnt/temp_disk
```

对块设备驱动进行调用测试。可以进入 /mnt/temp_disk 目录，并尝试复制系统 python 文件到该目录下，以验证是否可以使用该设备。

✧ # cp /usr/bin/python3 /mnt/temp_disk/表示将另一个目录中的文件复制到挂载目录下。

✧ # ls /mnt/temp_disk/表示查看复制进来的文件。

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ ls /mnt/temp_disk
/
alsa-utils        dbus              openvpn           spice-vdagent
anacron           gdm3              plymouth          sssd
apparmor          grub-common       plymouth-log      sysstat
apport            kerneloops        procps            ufw
bluetooth         keyboard-setup.sh python3           unattended-upgrades
console-setup.sh  kmod              rsync             uuidd
cron              lost+found        saned             whoopsie
cups              open-vm-tools     speech-dispatcher x11-common
```

 # df -h 表示查看资源使用情况，可以看到**新增**的文件系统的资源使用情况为 33%。

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo cp /usr/bin/
python3 /mnt/temp_disk/
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ df -h
文件系统           大小  已用  可用 已用% 挂载点
tmpfs              1.3G  2.2M  1.3G    1% /run
/dev/sda2           79G   58G   18G   77% /
tmpfs              6.4G   54M  6.3G    1% /dev/shm
tmpfs              5.0M  8.0K  5.0M    1% /run/lock
tmpfs              1.3G  140K  1.3G    1% /run/user/1000
/dev/sr1           6.0G  6.0G     0  100% /media/tt/Ubuntu 24.04.3 LTS amd64
/dev/sr0            96M   96M     0  100% /media/tt/CDROM
/dev/simp_blkdev    26M  7.9M   16M   33% /mnt/temp_disk
```

接着，我执行写操作（echo ... > ...）将一段文本写入该磁盘中的文件，随后执行读操作（cat）读取该文件。结果显示读取内容与写入内容一致，证明驱动程序的请求处理函数（queue_rq）能够正确地在内存和系统缓冲区之间传输数据，**读写功能正常**。

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo sh -c "echo 'Hello, World! Jintao Zhu' > /mnt/t
emp_disk/test.txt"
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ cat /mnt/temp_disk/test.txt
Hello, World! Jintao Zhu
```

在信息中**加入了我的姓名，以作区分**。

6. 卸载模块与清理

测试完成后，我使用 sudo umount 命令取消了设备的挂载，并使用 sudo rmmod simp_blkdev 命令卸载了驱动模块。

再次查看 lsblk 确认设备已消失，并通过 dmesg 查看内核日志，验证了模块卸载函数被正确调用，清理工作执行完毕 ！

```
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo umount /mnt/temp_disk
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo rmmod simp_blkdev
tt@tt-VMware-Virtual-Platform:~/下载/2327406014_朱金涛_实验十_源码$ sudo dmesg | tail
[  707.279423] simp_blkdev: loading out-of-tree module taints kernel.
[  707.280232] simp_blkdev: module verification failed: signature and/or required key missing - tainting kernel
[  707.417008] Simp Block Device Driver Loaded
[  709.366176] kauditd_printk_skb: 149 callbacks suppressed
[  709.366189] audit: type=1400 audit(1764572440.593:161): apparmor="DENIED" operation="open" class="file" profile="snap
.firmware-updater.firmware-notifier" name="/proc/sys/vm/max_map_count" pid=5223 comm="firmware-notifi" requested_mask="r
" denied_mask="r" fsuid=1000 ouid=0
[  830.116223] workqueue: blk_mq_run_work_fn hogged CPU for >10000us 11 times, consider switching to WQ_UNBOUND
[  893.286755] EXT4-fs (simp_blkdev): mounted filesystem 6b6ccc9b-a446-43f3-adea-89e3c0d77d30 r/w with ordered data mode
. Quota mode: none.
[  895.512816] workqueue: psi_avgs_work hogged CPU for >10000us 11 times, consider switching to WQ_UNBOUND
[ 1065.036785] EXT4-fs (simp_blkdev): unmounting filesystem 6b6ccc9b-a446-43f3-adea-89e3c0d77d30.
[ 1071.625088] Simp Block Device Driver Unloaded
```

四、 实验总结

1. 思考题：字符设备与块设备驱动程序的异同点分析

(1) 相同点:

- **内核模块架构**：两者都遵循 Linux 内核模块的基本框架，都需要编写加载函数和卸载函数，并且都需要在加载时向内核注册设备，在卸载时注销释放资源。
- **设备标识**：两者都使用“主设备号”和“次设备号”来标识设备。主设备号用于匹配驱动程序，次设备号用于区分同类设备中的个体。
- **用户空间接口**：在 Linux “一切皆文件”的设计哲学下，两者在用户空间都表现为 /dev 目录下的设备文件，用户应用程序都可以通过标准的系统调用（如 open, read, write, close）来访问它们。

(2) 不同点:

- **数据传输单位与方式**:
  - **字符设备**：以字节为单位进行传输，通过字符流的方式顺序访问，通常**不支持随机访问**（或者说随机访问效率很低），如串口、键盘。
  - **块设备**：以**块（Block，通常是扇区，如 512 字节或 4KB）为单位进行传输，支持随机访问**。在本实验中，我们定义的扇区大小即体现了这一点。
- **核心数据结构与接口**:
  - **字符设备**：核心结构是 cdev，主要实现 file_operations 结构体中的 read、write 等函数。用户调用 read，内核直接调用驱动的 read。
  - **块设备**：核心结构是 gendisk 。驱动程序不直接提供 read/write 接口给用户，而是提供**请求处理函数**（如实验中的 simp_blkdev_queue_rq ）。内核通过请求队列与驱动通信，驱动负责从队列中取出请求并处理。
- **缓冲与缓存机制（I/O 调度）**:

- **字符设备**：通常是**无缓冲**的直接读写，数据直接在用户空间和设备之间传输，实时性较强。
- **块设备**：高度依赖内核的**缓冲区/页高速缓存**。系统为了提高磁盘读写效率，会将多个读写请求进行合并、排序（电梯算法）等优化操作，放入请求队列中，然后再交给驱动程序处理。实验中初始化的 blk-mq（多队列）标签集合正是为了适配这种复杂的调度机制。

- **复杂程度**：
  - **字符设备**：实现相对简单，逻辑直观。
  - **块设备**：实现较为复杂。驱动不仅要处理硬件操作，还要适配内核的块设备层 API，处理请求队列、分散/聚集 DMA 传输等。本实验中为了适配新内核 API 进行的大量修改（如 blk_mq_alloc_disk 等）也印证了块设备驱动对内核机制的依赖更深。

2. 实验感悟：

本次实验中，我成功编写并验证了一个基于内存的虚拟块设备驱动程序，实现了一套 32MB 的虚拟磁盘系统 。

在实验实施过程中，我面临的主要挑战是 Linux 6.17.2 新内核带来的块设备层 API 重构。为此，我深入分析了内核差异并对驱动源码进行了关键适配：将头文件调整为 <linux/blkdev.h>，移除了已被废弃的 BLK_MQ_F_SHOULD_MERGE 标志位，并正确补全了 blk_mq_alloc_disk 函数的参数，从而成功解决了编译错误 。

在功能验证阶段，我通过编写核心的请求处理函数，实现了基于 memcpy 的内存读写逻辑，模拟了磁盘的数据传输过程 。随后，我严格按照编译、加载（insmod）、格式化（mkfs.ext4）、挂载及读写测试的流程进行操作。测试结果显示，虚拟设备能够被系统正

确识别（lsblk），且文件读写内容一致，证明了驱动程序的稳定性和正确性 。

通过本次实验，我不仅掌握了 Linux 块设备驱动的基本编写框架和 Makefile 的构建规则，更深刻理解了 blk-mq 多队列架构的工作机制以及内核模块在实际操作系统中的管理与调用流程。